I dOn'T (Z)Know: An Architecture for Zero-Knowledge Cross-Platform IoT Applications

Ekene Attoh[1][0000−0002−8590−1005] and Beat Signer[1][0000−0001−9916−0837]

Vrije Universiteit Brussel, Pleinlaan 2, Brussels, Belgium
{eattoh, bsigner}@vub.be

Abstract. The Internet of Things (IoT) fosters connected environments where devices interact with one another and with users to enable context-aware applications. End-user authoring tools empower individuals to create personalised automations, such as health-related rules that respond to physiological metrics. However, these tools are often tied to specific vendors, limiting the portability of user-defined automations across platforms. This restriction poses significant challenges in domains like healthcare, where users may depend on such automations for daily assistance. To address this issue, in our recent research, we have proposed a *write once, run anywhere* paradigm to enable rule portability across heterogeneous IoT environments. While this approach improves continuity, it also raises privacy concerns, as user data may be exposed during the migration of automations between platforms. In this paper, we address some of these privacy challenges by introducing a representative user scenario, analysing related work and proposing a privacy-preserving IoT architecture (IOT-ZK) that makes use of zero-knowledge proofs, along with a proof-of-concept implementation. Our proposed solution supports secure and portable automation across IoT platforms, with particular emphasis on safeguarding user data in sensitive domains such as healthcare.

Keywords: zero-knowledge IoT, end-user privacy, cross-platform IoT

1 Introduction and Background

Lieberman et al. [16] define *end-user development* as "*a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artefact*" to develop and adapt systems according to their individual backgrounds and skill levels. Various tools have been proposed to support end-user IoT development, including those presented in [9,8,7,28,24]. As noted by Markopoulos et al. [18], one of the most common programmatic mechanisms for end-user control in IoT applications is rule specification. A prominent example of a rule-based solution is the *If This Then That* (IFTTT)[1] platform, which allows users to define actions triggered by specific conditions. For instance, a user might configure a rule such as "*If my smart sensor detects that my resting*

[1] https://ifttt.com

heart rate is below 40 bpm, then change the colour of my smart lamp to red", enabling them to receive notifications in their home environment whenever a health metric deviates from the norm.

Previous studies have also shown that rule-based approaches are generally intuitive and accessible for end users [4,25]. However, a significant limitation of many current end-user development tools is their implicit vendor lock-in. This implies that when users create personalised IoT applications using these tools, they are often only usable on those platforms and on the specific devices for which they were created. In fact, Corno et al. [5] stated that most end-user development platforms adopt a vendor-centric abstraction, requiring each online service to be programmed in a proprietary manner. Similarly, Noura et al. [19] emphasised that most IoT solutions lack support for *cross-platform* and *cross-domain* interoperability. In our own research, we argue that resolving interoperability issues is key to freeing users from vendor-specific constraints [2] and proposed a *write once, run anywhere* approach for authoring end-user IoT applications [3]. This approach allows end users to retain their preferred authoring tools while enabling their automations to run across different platforms.

Our solution translates a user's rules into a high-level representation that can then be executed in any (new) environment on any platform that implements an EUPont runtime (rule engine). For example, the previous rule "*If my smart sensor detects that my resting heart rate is below 40 bpm, then change the colour of my smart lamp to red*" could be configured for a specific sensor and smart lamp in the user's home. This means that if the user went on holiday to a hotel that offers an IoT platform with a similar sensor and smart lamp but from a manufacturer not supported by their home platform, their automation would no longer work. Using the solution proposed in [3], the rule can be expressed as "*If received from device, then set lighting*". This abstraction allows the trigger to be mapped to any compatible heart rate sensor and the action to any smart lamp (the sensor and lamp present in the hotel), regardless of manufacturer, rather than only to devices supported by the user's home platform. The high-level representation is based on the EUPont abstraction proposed by Corno et al. [6], enabling the execution of rules on any EUPont-compliant platform. While this cross-platform interoperability offers great flexibility, it also introduces new challenges, particularly regarding privacy. Since, thanks to our solution proposed in [3], a user can now use their automation in environments other than their home, the *foreign* environment might gain knowledge about the user's resting heart rate values and what they consider an abnormal value. In critical domains, such as healthcare, users may be better protected against potential data misuse when personal data is controlled in a certain way. As noted by Ziegeldorf et al. [30], IoT solutions often contain sensitive information that may be of interest to third parties. For example, information about a user's heart rate, like in our example, could be exploited by insurance companies to increase their premiums or even deny a new contract [12]. Magara and Zhou [17] identified user profiling and third-party data sharing as privacy concerns in IoT. As Roman et al. [22] put it, the data deluge caused by billions of entities creating information poses a threat to privacy and

users must thus have tools that allow them to retain their anonymity in this connected world. They add that IoT must adopt privacy by design, providing user-centric support for security and privacy from its very foundations. Furthermore, they note that entities do not need to provide all the data they produce, but only the data required by external entities for a particular service [22]. The issue of trust is also discussed by Roman et al. [22], where uncertainty about future actions that all components may take must be addressed. Returning to our example, after sharing their rule with a platform in a foreign environment, users may be left uninformed about the future of their data once they leave that environment.

To address the highlighted privacy issues, in the next section we present our user scenario and analyse related work. Based on these, we introduce and describe our architecture, aiming to address the problem and present a proof-of-concept implementation of the architectural component.

2 Related Work

Zero-knowledge proof systems were presented in [11] where the researchers defined that a proof system is considered zero knowledge if whatever the *verifier* can compute while interacting with the *prover*, it can compute by itself without going through the *protocol*. In simple terms, this means that whatever the verifier learnt from talking with the prover, they could have learnt on their own without talking to the prover. The prover is the party that has (secret) information and wants to convince another party that the information satisfies a certain property. On the other hand, the verifier is the party that checks the validity of the claim without learning anything about the information itself. Goldreich and Oren [10] defined a zero-knowledge proof as anything a verifier could determine while interacting with the prover; given access to certain information, it could also be determined independently by the verifier using that same information, without interacting with the prover. *Zero Knowledge Proofs* (ZKPs) are therefore techniques to verify claims without revealing the information itself, with the prover sharing proof of their claim with the verifier, who then verifies the accuracy of the proof without learning any additional information [1]. A valid ZKP system must satisfy the following three criteria [10].

- **Completeness** – If the statement is true, an honest prover can convince an honest verifier of its validity.
- **Soundness** – If the statement is false, no dishonest prover can convince an honest verifier that it is true.
- **Zero Knowledge** – If the statement is true, the verifier learns nothing about the secret information itself, only that the claim is true.

ZKPs can be interactive, requiring multiple exchanges, or non-interactive, with verification occurring in a single step. Non-interactive forms, like *zkSNARKs* (Zero-Knowledge Succinct Non-interactive Arguments of Knowledge), are concise, storage-efficient and allow computed results to serve as statements. Unlike

interactive ZKPs, they replace the verifier's challenges with a shared reference value, allowing proof sharing with third parties [1]. As stated by Reitwiessner [21], zkSNARKs are impressive; one can verify the correctness of computations without having to execute them, and one will not even learn what was executed. They explain the properties of zkSNARKS as follows:

- **zk** – during the interaction, the verifier learns nothing except the validity of the statement.
- **Succinct** – proof sizes are tiny compared to the underlying computation.
- **Non-interactive** – after an initial setup, the prover sends a single proof that anyone can verify without further interaction.
- **Arguments** – "*the verifier is only protected against computationally limited provers. Provers with enough computational power can create proofs/arguments about wrong statements (note that with enough computational power, any public-key encryption can be broken). This is also called computational soundness, as opposed to perfect soundness*".
- **of Knowledge** – a proof can only be made if the prover knows the required secret (witness).

In the literature, ZKPs have been proposed for IoT. For example, in the context of device authentication, Zhong et al. [29] proposed an efficient, secure and on-demand communication protocol using zero-knowledge proofs (ZKPs) that allows the prover to provide evidence of its secret without revealing it to the verifier. One way to verify a device's authenticity is to use physically unclonable functions (PUFs) as a unique device fingerprint. Therefore, with this solution, the edge device (the prover) convinces the central server (the verifier) of the unique PUF response stored on the device without requiring the actual storage of PUF responses on the server. Given the limited computational resources and the potential for limited network connectivity prevalent in IoT, the research Walshe et al. [27] proposed an authentication protocol based on non-interactive zero-knowledge proofs (NIZKPs). Traditional authentication approaches often require storing authentication data on devices, creating security risks if the device is compromised, or they rely on Public Key Infrastructure (PKI)-based solutions that impose high computational and maintenance overheads.

In the proposed protocol, the setup phase generates a *Merkle tree* [15] whose leaves represent the hashed device authentication data. A Merkle tree is a binary tree in which each leaf node contains the hash of a data block, and each non-leaf node contains the cryptographic hash of the concatenation of its child nodes. In blockchain systems, Merkle trees efficiently summarise all transactions in a block, with each transaction being a leaf node. This structure allows users to verify whether a particular transaction is included in a block without downloading the entire blockchain. For example, in Bitcoin, Merkle trees are used to create a compact summary of all transactions in a block, enabling lightweight nodes to verify transactions independently [15]. The prover thus generates a proof by presenting the hash path from a leaf node to the Merkle root. The verifier checks this proof against the stored Merkle root without accessing sensitive authentication data. This method ensures that credentials are not stored in plain text, is

resistant to replay and man-in-the-middle attacks, and functions effectively under reduced connectivity conditions [27]. Ramezan and Meama [20] proposed a framework that enhances IoT security by using Zero-Knowledge Proofs (ZKPs) and blockchain to ensure the integrity of firmware execution and data processing, even in untrusted or compromised devices. Within the framework, every IoT device is equipped with a ZKP generator to validate the integrity of the programs it executes. The zk-device ensures that its output originates from a circuit designed by the manufacturer, certifying that the IoT device operates exactly as intended by its manufacturer and guarantees that both the program and its inputs remain unchanged, resulting in reliable and trustworthy output. The zk-devices also implement a ZKP scheme to verify a chain of computations across a series of IoT devices.

To enable the use of user-defined automations in a new environment, the environment must have access to a user's rules. In [3], we introduced the use of *Solid pods* [23], enabling users to store their data in a location (pod) that belongs to them and grant applications access to the data. Assuming a user's rules have been translated into the high-level representation described in [3] and stored in their Solid pod, they can grant the hotel's platform access to their rules.

Referring to our earlier example where a user goes on holiday and stays in a hotel, the hotel can thus execute their rules by implementing an EUPont runtime (rule engine) as described in [6]. However, this poses a privacy risk, as a user may not wish to disclose the thresholds defined in their rules to the hotel. In the next section, we therefore introduce an IOT-ZK architecture using zero-knowledge proofs in end-user IoT applications, enabling data sharing across different platforms without revealing sensitive information. To the best of our knowledge, this approach has not previously been explored in the literature.

3 IOT-ZK Architecture

The ageing of the population is a global phenomenon with far-reaching consequences, presenting challenges and opportunities for global healthcare systems [14,13]. End-user IoT solutions can help users maintain or even improve their quality of life. We have therefore developed the following end-user IoT scenario to set the context for the problem we seek to highlight and address in this paper. Mildred, a 60-year-old pensioner with health conditions, uses *IoT Platform A*, which provides the If This Then That (IFTTT) method to automate aspects of her daily life. She has created some rules to remind her to monitor her insulin levels, do her exercises and manage her evening relaxation routine. For example, she defined the following rules:

- If my smart sensor detects that my insulin level is below threshold y, then change the colour of the lamp in the kitchen to red.
- If the time is 11:00 am, then start playing a workout song on my smart speakers
- If the time is 19:00 pm, then set the thermostat to 21 degrees Celsius

These rules are stored on *IoT Platform A* and help Mildred maintain her well-being. Later, Mildred travels abroad and stays in a hotel with some smart devices. She expects her automations to continue working in the hotel. However, none of her rules works, because the hotel uses a different IoT platform *(IoT Platform B)* that neither has access to her rules nor uses the *IFTTT* programming model and a user interface she is familiar with. Mildred also does not want to disclose the specific trigger conditions of her personal rules to the hotel platform—such as her insulin level threshold, the time she starts working out and the time she starts heating to her preferred temperature—because she considers these to be private information. Therefore, our problem statement is: "*How can users benefit from their IoT automations in new environments without sharing specific details about their rules?*" In the following, we introduce the components to address this research question.

We propose the use of *zero-knowledge proofs (ZKPs)* in a new cross-platform IoT architecture (IOT-ZK) to cryptographically hide the details of a user's rules that they wish to hide from *untrusted* platforms; that is, platforms in new (foreign) environments. Zero knowledge proofs follow three steps:

- In the ***commitment*** step, a prover commits to a secret value without revealing it. It is like sealing a secret in a box and showing that box to interested parties, demonstrating that it contains a value without revealing the value. This step ensures that the secret value is bound to a unique cryptographic commitment while remaining hidden from any third parties. Note that the secret value cannot be changed once it is bound and a new commitment will have to be generated for a new secret value. We therefore introduce functionality that will create a commitment to the details of rules which users wish to hide from untrusted platforms. In our example scenario, commitments will be created for *insulin threshold value*, *11:00 am* and *19:00 pm*. In Listing 1.1, we highlight the trigger for Mildred's *set thermostat* rule, with the commitment value (instead of 19:00 pm) in the `eupont:details` field. It shows a snippet of the cryptographic data generated in the commitment step. This data hides the secret value 19:00 pm by encoding it into a cryptographic commitment.
- In the ***proof generation*** step, the prover uses the secret value bound to the previously generated commitment to produce a proof that this value satisfies a condition (constraint). A proof is a piece of cryptographic data that convinces another party that a certain statement about a secret is true, without revealing the secret itself. The proof generation step constructs an arithmetic circuit that encodes a logical statement about the secret value bound to the commitment. In our example, this circuit encodes the condition that the secret value is equal to 19:00 pm. Formally, this relation can be represented as $f(s\_value, p\_value)$, where f denotes the operator implemented by the circuit, s_value is the value only known to the prover and p_value is a public value. In this example, f represents an *equality* check, which indicates that the secret value bound to the commitment is equal to the given public value. Hence, we introduce some functionality that can

generate such proofs $f(s\_value, p\_value)$. For our scenario, proofs will be generated and stored for the three values that Mildred wishes to hide from the hotel platform.

– In the ***verification*** step, the verifier uses the public commitment, the received proof and a known public input (e.g. a sensor value) to check whether the proof is valid with respect to the same arithmetic circuit used in proof generation. The verifier does not learn the secret itself; rather, it becomes convinced that the secret value bound to the commitment satisfies the condition encoded in the proof. Thus, a successful verification provides cryptographic assurance that the prover's claim is true without revealing the underlying secret.

```
# Trigger definition
ex:applet_1_set_thermostat_trigger a eupont:EveryDayTrigger ;
eupont:triggers ex:applet_1_set_thermostat ;
eupont:details b'geppetri_datablock [
   0x4538b762caaa863ce9ffc68788b708347fd4a8ebe096681bd08a

...

8a5d75e88baf10c9c158,0x21174aeb5cd30647d5950660336ffa65d154402f629a0b312f0d6330df4c] ]\n' .
```

Listing 1.1. Example zero-knowledge rule

As an illustration, we take the rule "*If the time is 19:00 pm, then set the thermostat to 21 degrees Celsius*". The commitment step will generate a commitment value to cryptographically hide "*19:00 pm*" and the rule can be rewritten as "*If time is <commitment value>, then increase temperature*". Note that "*increase temperature*" is used instead of "*set the thermostat to 21 degrees Celsius*" because the rule is first translated to the high-level EUPont abstraction. A proof is then generated using the secret value bound to the commitment, together with a public input (19:00 pm), and the function $f(s\_value, 19:00\,pm)$ where f represents an *equality* check. This therefore generates a proof that encodes the condition that the secret value bound to the commitment satisfies an equality relation with respect to the given public value (19:00 pm). An example of such a proof, which contains a snippet of the cryptographic data generated in the proof generation step, is shown in Listing 1.2. This data therefore encodes the equality condition between the secret value bound to the commitment and the public value (19:00 pm in our example).

```
b'geppetri_qapproof [ \n  0xc00d6ae0e7ea78205d464b412e8f_0x306073a49e7088840
cb33992f391b32db4efa97add480b20ecb57af55fd576cf\

...

e36f5af5ee2320_0x56a2851119e4ad22917b6fada5032524245b8af9ea323 ]\n]\n'
```

Listing 1.2. Example zero-knowledge proof

The rule bearing the commitment value and the proof is then stored and made available for retrieval by the hotel platform. The hotel platform can read the rule with its commitment value as well as the proof from Mildred's untrusted pod. It can then perform a verification $v(proof, commitment, p\_value)$ where p_value

represents the *current time* obtained from a time sensor (e.g. a digital clock). The verification checks whether the proof is valid for the given commitment and public input. If this verification succeeds, the hotel's platform is convinced that the hidden value bound to the commitment satisfies the equality condition with respect to the supplied time. The hotel platform will then trigger the rule's action. With this setup, Mildred retains full control over her personal data because the specific values of her rule conditions are not directly shared with the hotel platform, thereby preserving her privacy. An illustration of this flow is provided in Fig. 1.

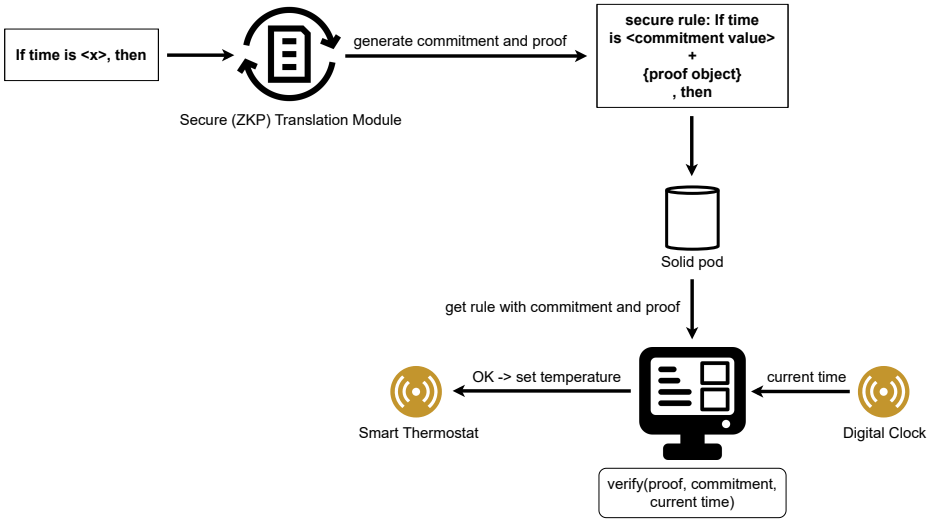

Fig. 1. IoT zero-knowledge illustration

The zero-knowledge proofs are implemented using the PySNARK[2] Python library. PySNARK enables one to program zk-SNARKs (verifiable computations) in Python 3. It allows proofs to refer to committed data using Geppetri [26] and has three main applications:

- It allows proofs to refer to external private inputs from parties other than the trusted third party.
- It allows different verifiable computations to share secret data.
- It allows a verifiable computation to be divided into multiple subcomputations, each with its own evaluation and verification keys.

Therefore, in order to allow users to not only benefit from being able to execute their rules across platforms and in new environments, but to also maintain their privacy in those new environments, we introduce a zero-knowledge proof extension to our earlier proposed architecture [3] as illustrated in Fig. 2.

[2] https://github.com/meilof/pysnark

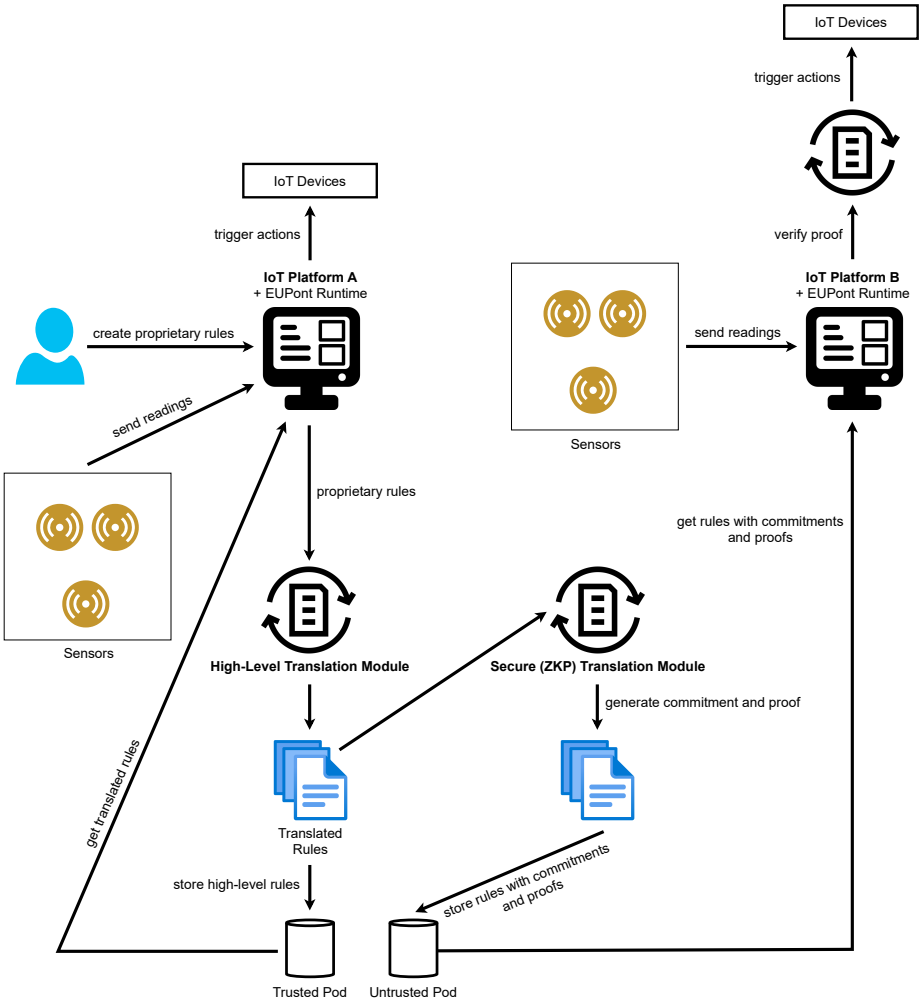

Fig. 2. Cross-platform IoT architecture with ZKP

For her trusted home platform, Mildred therefore creates the `Trusted Pod` where rules with full details are stored, while rules with their details hidden are stored in the `Untrusted Pod`. Finally, the foreign platform can read the rules from the *Untrusted Pod* and verify whether the rule's action can eventually be triggered. In Listing 1.3, we see a Python code example of this flow. While Mildred is in a hotel during her vacation, the presented IOT-ZK architecture enables her to benefit from the same rules she has in her home while preserving her privacy by concealing the details of those rules from the hotel's IoT system.

```
# convert Mildred's IFTTT rules to EUPONT representation
turtle_rule = convert_to_eupont_representation(rule=combined_method_translation)

turtle_filename = save_file(filename="<FILEPATH>".format(
  rule_name=combined_method_translation["rule_name"]), content=turtle_rule)

# save the converted rule in solid pod for trusted platforms with all the details
add_to_solid_pod(pod_url="<POD_URL>", file_name="{rule_name}-mildred.ifttt.ttl".format(
  rule_name=combined_method_translation["rule_name"]),
    file_content=turtle_rule, content_type="application/ttl")

# convert Mildred's IFTTT rules to EUPONT representation with zk commitment and proof
zk_turtle_rule, zk_turtle_file_bytes = convert_to_zk_eupont_representation(
  rule=combined_method_translation)
zk_turtle_filename = save_file(filename="<FILEPATH>/{rule_name}-mildred.zk.ifttt.ttl".
  format(rule_name=combined_method_translation["rule_name"]), content=zk_turtle_rule)

# save the converted rule in solid pod for untrusted platforms with zk commitment
add_to_solid_pod(pod_url="<POD_URL>",
  file_name="{rule_name}-mildred.zk.ifttt.ttl".format(rule_name=
  combined_method_translation["rule_name"]),
    file_content=zk_turtle_rule, content_type="application/ttl")

# simulate Hotel Platform proof verification
is_valid_proof = verify_proof(comm_name=zk_turtle_file_bytes[0][0], comm_bytes=
  zk_turtle_file_bytes[0][1], proof_bytes=zk_turtle_file_bytes[1],
    k=zk_turtle_file_bytes[0][2])

# simulate Hotel Platform executing rule action
if is_valid_proof:
  print("Executing rule action: " + combined_method_translation["translated_action"])

## EXAMPLE OUTPUT
Proof generated successfully
Proof is valid: True
Executing rule action: SetTemperatureAction
```

Listing 1.3. Zero-knowledge example implementation

The `Secure (ZKP) Translation Module` will perform the commitment as well as proof generation steps, and store the results of those steps. We propose that the translated rules, along with commitments and proofs, be stored in a dedicated Solid pod as shown in Figure 2. This will ensure that users can grant untrusted platforms only access to the pod that stores the rules with commitments and proofs, while their trusted platforms can have access to a different pod containing their rules with full details.

4 Conclusion and Future Work

With the increasing proliferation of end-user IoT solutions and the potential for cross-platform interoperability as proposed in our earlier research [3], new privacy concerns have emerged. These concerns stem from the possibility that unfamiliar IoT platforms may gain access to sensitive user information embedded in automation rules. To address this problem, we propose an extension of our previously introduced architecture [3] by augmenting the translation module with zero-knowledge features. This enhancement allows users to conceal the specific details of their rules from third-party platforms. As a result, an unfamiliar

platform can still access and execute a user's rule without learning the exact conditions that trigger it. This capability opens up a range of compelling use cases, including the scenario presented in this paper. It is also particularly relevant in shared IoT environments, such as offices or smart cities, where multiple users interact with common platforms and devices. While we have presented a prototype implementation of our proposed solution, future work will focus on refining this system and conducting user studies to gather feedback for further development. We are aware that in our example, although Mildred's rules are never shared with the hotel's platform, the platform may still infer the secret value via a so-called *inference attack* by observing when the rule is triggered (e.g. when the thermostat is activated at 19:00 pm). Therefore, we plan to further investigate this issue and propose some mitigation strategies. Note that in our scenario, Mildred has a rule that retrieves her insulin levels and takes action based on a threshold she has set. In this case, the regular readings from her insulin monitor are as sensitive as the threshold value she defines as being low. While we currently only hide rule-specific details, in future work, we plan to investigate using similar zero-knowledge proofs to also protect sensor readings from unfamiliar third-party platforms.

References

1. Aad, I.: Zero-Knowledge Proof. In: Trends in Data Protection and Encryption Technologies. Springer Nature Switzerland (2023). https://doi.org/10.1007/978-3-031-33386-6_6
2. Attoh, E., Signer, B.: A Middleware for Implicit Human-Computer Interaction Across IoT Platforms. In: Adjunct Proceedings of UbiComp 2021, International Joint Conference on Pervasive and Ubiquitous Computing. Virtual Conference (September 2021). https://doi.org/10.1145/3460418.3479311
3. Attoh, E., Signer, B.: Towards a Write Once Run Anywhere Approach in End-User IoT Development. In: Proceedings of IoTBDS 2024, 9th International Conference on Internet of Things, Big Data and Security. Angers, France (April 2024). https://doi.org/10.5220/0012675800003705
4. Cabitza, F., Fogli, D., Lanzilotti, R., Piccinno, A.: Rule-based Tools for the Configuration of Ambient Intelligence Systems: A Comparative User Study. MTAP **76** (February 2017). https://doi.org/10.1007/s11042-016-3511-2
5. Corno, F., De Russis, L., Monge Roffarello, A.: Devices, Information, and People: Abstracting the Internet of Things for End-User Personalization. In: Proceedings of IS-EUD 2021, 8th International Symposium on End-User Development. Online Conference (July 2021). https://doi.org/10.1007/978-3-030-79840-6_5
6. Corno, F., De Russis, L., Roffarello, A.M.: A High-Level Semantic Approach to End-User Development in the Internet of Things. International Journal of Human-Computer Studies **125** (May 2019). https://doi.org/10.1016/j.ijhcs.2018.12.008
7. Coutaz, J., Crowley, J.L.: A First-Person Experience with End-User Development for Smart Homes. IEEE Pervasive Computing **15**(2) (April–June 2016). https://doi.org/10.1109/MPRV.2016.24

8. Desolda, G., Ardito, C., Matera, M.: End-User Development for the Internet of Things: EFESTO and the 5W Composition Paradigm. In: Rapid Mashup Development Tools (2017). https://doi.org/10.1007/978-3-319-53174-8_5
9. Dey, A.K., Hamid, R., Beckmann, C., Li, I., Hsu, D.: a CAPpella: Programming by Demonstration of Context-aware Applications. In: Proceedings of CHI 2004, SIGCHI Conference on Human Factors in Computing Systems. Vienna, Austria (April 2004). https://doi.org/10.1145/985692.985697
10. Goldreich, O., Oren, Y.: Definitions and Properties of Zero-Knowledge Proof Systems. Journal of Cryptology **7**(1) (1994). https://doi.org/10.1007/BF00195207
11. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof-Systems. In: Proceedings of STOC 1985, 17th Annual ACM Symposium on Theory of Computing. Providence, USA (May 1985). https://doi.org/10.1145/22145.22178
12. Henze, M., Hermerschmidt, L., Kerpen, D., Häußling, R., Rumpe, B., Wehrle, K.: User-driven Privacy Enforcement for Cloud-based Services in the Internet of Things. In: Proceedings of FiCloud 2024, 2nd International Conference on Future Internet of Things and Cloud. Barcelona, Spain (August 2014). https://doi.org/10.1109/FiCloud.2014.38
13. Islam, Q.: Innovation in Primary Healthcare in the Twenty-First Century. Journal of Health Management **23**(1) (2021). https://doi.org/10.1177/0972063421994987
14. Jane Osareme, O., Muonde, M., Maduka, C.P., Olorunsogo, T.O., Omotayo, O.: Demographic Shifts and Healthcare: A Review of Aging Populations and Systemic Challenges. International Journal of Science and Research Archive **11** (2024). https://doi.org/10.30574/ijsra.2024.11.1.0067
15. Kuznetsov, O., Rusnak, A., Yezhov, A., Kuznetsova, K., Kanonik, D., Domin, O.: Merkle Trees in Blockchain: A Study of Collision Probability and Security Implications. Internet of Things **26** (2024). https://doi.org/10.1016/j.iot.2024.101193
16. Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-User Development: An Emerging Paradigm. In: End User Development (2006). https://doi.org/10.1007/1-4020-5386-X_1
17. Magara, T., Zhou, Y.: Internet of Things (IoT) of Smart Homes: Privacy and Security. Journal of Electrical and Computer Engineering **2024**(1) (2024). https://doi.org/10.1155/2024/7716956
18. Markopoulos, P., Nichols, J., Paternò, F., Pipek, V.: End-User Development for the Internet of Things. ACM Transactions on Computer-Human Interaction (TOCHI) **24**(2) (2017). https://doi.org/10.1145/3054765
19. Noura, M., Atiquzzaman, M., Gaedke, M.: Interoperability in Internet of Things: Taxonomies and Open Challenges. Mobile Networks and Applications **24**(3) (2019). https://doi.org/10.1007/s11036-018-1089-9
20. Ramezan, G., Meamari, E.: zk-IoT: Securing the Internet of Things With Zero-Knowledge Proofs on Blockchain Platforms. In: Proceedings of ICBC 2024, IEEE International Conference on Blockchain and Cryptocurrency. Dublin, Ireland (May 2024). https://doi.org/10.1109/ICBC59979.2024.10634342
21. Reitwiessner, C.: zkSNARKs in a Nutshell. Ethereum Foundation Blog (December 2016), https://blog.ethereum.org/2016/12/05/zksnarks-in-a-nutshell
22. Roman, R., Zhou, J., Lopez, J.: On the Features and Challenges of Security and Privacy in Distributed Internet of Things. Computer Networks **57**(10) (2013). https://doi.org/10.1016/j.comnet.2012.12.018

23. Sambra, A.V., Mansour, E., Hawke, S., Zereba, M., Greco, N., Ghanem, A., Zagidulin, D., Aboulnaga, A., Berners-Lee, T.: Solid: A Platform for Decentralized Social Applications Based on Linked Data. Tech. rep., MIT CSAIL & Qatar Computing Research Institute (2016), https://cs.brown.edu/courses/csci2390/2021/readings/solid.pdf
24. Sanctorum, A., Signer, B.: eSPACE: Leveraging Theoretical Foundations for the End-User Development of Cross-Device and IoT Applications. ACM Transactions on Computer-Human Interaction (TOCHI) **32**(3) (2025). https://doi.org/10.1145/3716133
25. Ur, B., Pak Yong Ho, M., Brawner, S., Lee, J., Mennicken, S., Picard, N., Schulze, D., Littman, M.L.: Trigger-Action Programming in the Wild: An Analysis of 200,000 IFTTT Recipes. In: Proceedings of CHI 2016, SIGCHI Conference on Human Factors in Computing Systems. San Jose, USA (May 2016). https://doi.org/10.1145/2858036.2858556
26. Veeningen, M.: Pinocchio-based Adaptive zk-SNARKs and Secure/Correct Adaptive Function Evaluation. Cryptology ePrint Archive, Paper 2017/013 (2017), https://eprint.iacr.org/2017/013
27. Walshe, M., Epiphaniou, G., Al-Khateeb, H., Hammoudeh, M., Katos, V., Dehghantanha, A.: Non-Interactive Zero Knowledge Proofs for the Authentication of IoT Devices in Reduced Connectivity Environments. Ad Hoc Networks **95** (2019). https://doi.org/10.1016/j.adhoc.2019.101988
28. Wang, T., Qian, X., He, F., Hu, X., Huo, K., Cao, Y., Ramani, K.: CAPturAR: An augmented Reality Tool for Authoring Human-involved Context-aware Applications. In: Proceedings of UIST 2020, 33rd Annual ACM Symposium on User Interface Software and Technology. Virtual Conference (October 2020). https://doi.org/10.1145/3379337.3415815
29. Zhong, Y., Hovanes, J., Guin, U.: On-demand Device Authentication Using Zero-Knowledge Proofs for Smart Systems. In: Proceedings of GLSVLSI 2023, Great Lakes Symposium on VLSI 2023. Knoxville, USA (June 2023). https://doi.org/10.1145/3583781.3590275
30. Ziegeldorf, J.H., Morchon, O.G., Wehrle, K.: Privacy in the Internet of Things: Threats and Challenges. Security and Communication Networks **7**(12) (2014). https://doi.org/10.1002/sec.795